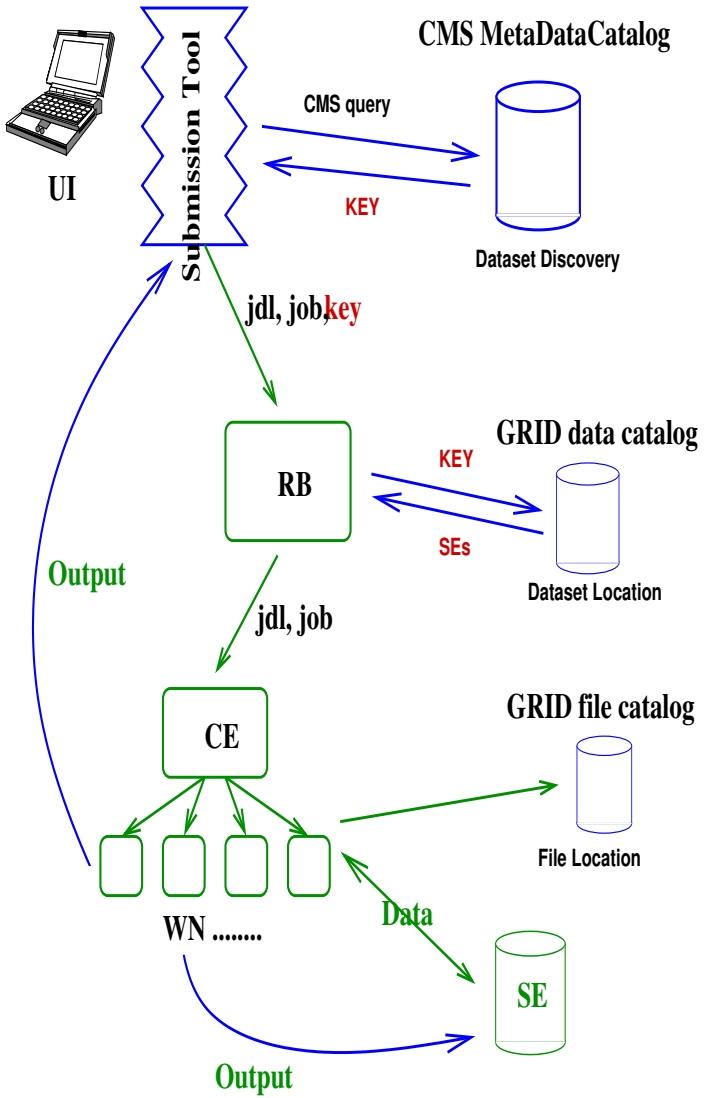
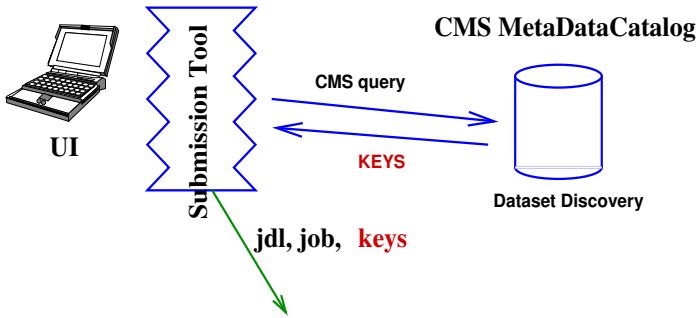
Catalogs: proposed architecture

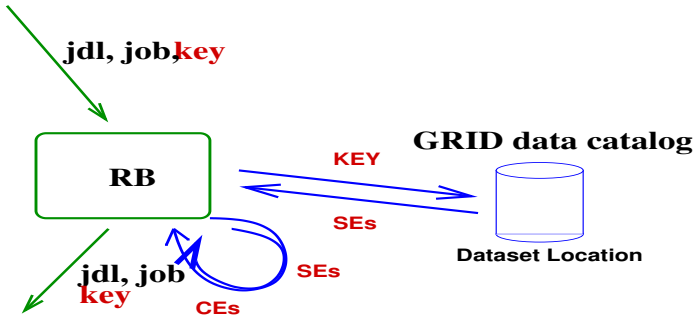


- Three level of catalogs, with defined responsibility and scope
  1. CMS specific Dataset Bookkeeping System
     user access point to next step: CMS responsibility, Centralized (replicated)
  2. Data Location Service
     should be Grid responsibility, **not** CMS, Global (replicated/distributed)
  3. Local File Catalog
     Grid responsibility, Local

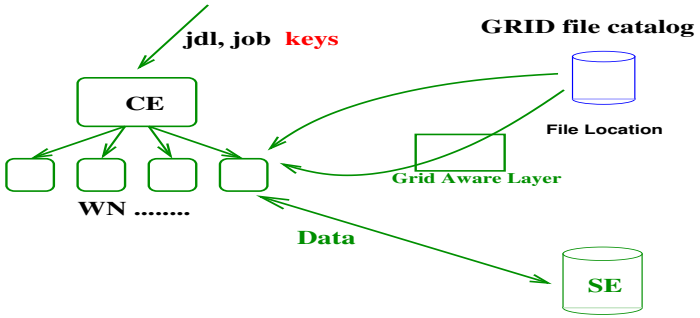


- CMS Dataset Bookkeeping System

- User (or user oriented tool CRAB) access point to Data Discovery and access
- Input user Query: any type, possibly Google like
- Must know about all available data, together with all Data attributes (sw version, calibration, detector condition, processing cards, etc. . .)
- Does not know about data location
- Return list of keys corresponding to Data Blocks
- Key list will be passed to next catalog



- Data Location Service
  - Grid responsibility
  - Accessed at Resource Broker level: Global
  - Accessible also from UI
  - Input is list of keys, corresponding to Data Blocks
  - `inputData = 'key1','key2',...`
  - Output is list of Storage Elements hosting Data Blocks
  - Not direct files knowledge is needed
  - Data Discovery is done using DB or collection of Data Blocks
  - RB finds CEs fine for SEs and choose CE
  - keys are (eventually) sent further down



- Local File Catalog
  - Accessed at CE/WN level: Local
  - Input is again list of keys, Data Blocks
  - Output is list of physical files corresponding to required Data Blocks
  - Directly used by COBRA, if POOL file catalog
  - Or transformed into POOL format by Grid aware layer (CRAB job wrapper)
  - If one POOL catalog (mysql) per site, do not need to extract Data Block fragment: COBRA application uses only what is needed

Feature and requirements

- Must answer to question: where is the data
- Data is defined in term of "Data Block", **not as list of files**
- Data Block Definition is fully experiment specific
- Data Block does correspond to a set of files, but this information is not needed at DLS level
- Data Block composition is static
- DLS get as input a set of keys which correspond to DataBlock, and should return the list of SE which contains this Data Block
- Interface could be DLI like (or anything else)
- Possible to query the service directly from UI

- A Data Block is "atomic": unbreakable. Don't need to deal with Data Block fragment
- No or very limited MetaData
- Definition of Data Block and attribute of Data Block are dealt with in the Dataset Bookkeeping System
- Eventually MetaData only for Data availability (such as: Data Block is on Tape, on Disk, pinned, pre-staged, etc . . .)
- Custodialship of the Data ("master" copy, never to be deleted)
- Data Block can be hierarchical: Dataset is made of Data Blocks, still under discussion
- Dataset composition can evolve

- Filling the DB is Experiment responsibility.
- When new data is created, an entry is added into Dataset Bookkeeping System describing all the attribute of the Data and the "key"
- Uniqueness of Keys is guaranteed by DBS
- To allow different scope for data (CMS, Analysis group, single person), could foresee a "namespace" for the key, eventually with a naming convention (CMS::key1, CMS:PRSMU:key2, etc. . .)
- When data is actually created, the initial location of Data Block is registered into the DLS
- Registration must be Transactional
- When data is replicated a new entry in the DLS is added by the data movement tool (Phedex) when the copy if complete and validated
- When data is removed the corresponding entry in the DLS is removed, then the data is removed.